Appendix C

Experimental setup

In order to accurately evaluate the performance of a classifier, one must use an appropriate evaluation protocol.

Since several of our experiments are applied to genomic datasets with few instances, a simple random split of the data into a training and a testing set was deemed inappropriate. Indeed, in those cases, the small size of the various testing sets would have resulted in unreliable measures of the classifier's performance.

There are several solutions to this problem such as repeated random split or cross-validation. Ultimately, we decided to use a cross-validation scheme as it guarantees that the classifier will be tested on all possible instances.

C.1 Double cross-validation

Since we are evaluating a classification algorithm and not a model with fixed hyper-parameter values, we use a double cross-validation scheme. This strategy consists of two cross-validations, one embedded inside the other. The inner cross-validation is used to tune the various hyper-parameter values while the outer one is used to evaluate the performance of each tuned model.

This protocol is sensible because it perfectly mimics what would happen in actual classification where the testing set's class labels are completely unknown. Indeed, for each iteration of the outer cross-validation:

1. The hyper-parameters are tuned on the training set using cross-validation.
2. A model is trained on the whole training set.
3. The model is used to predict the class labels for the testing set. The testing set is completely separate (i.e. not used to train or tune the model).

C.1.1 Hyper-parameter space

The algorithms considered in this report boast a large number of hyper-parameters[1]. This makes tuning over large ranges of hyper-parameters prohibitive from a temporal perspective, in particular for genomic datasets. Consequently, the results reported in this thesis were obtained using reduced ranges of hyper-parameters. Ranges were limited to a few reasonable values for the hyper-parameters and the resulting measures may not correspond to the best possible models on the various datasets.

[1] 7 plus the θ vector for *discRIT*. The number is comparable for *covRIT* and *relRIT*.

C.2 Classification scores

In order to evaluate the classification performance, we need to use an appropriate metric. The most obvious one is classification accuracy, which is defined as

$$acc = \frac{TP + TN}{TP + TN + FP + FN}$$

However, there is a problem with this particular metric: it doesn't always provide an accurate assessment of the classification performance when the dataset is unbalanced. When the class priors $P(C)$ are unbalanced, we may find ourselves with a classifier that has a good accuracy, say 95%, even though the classifier is trivial and only predicts the majority class.

To solve this problem, we will use another metric called the balanced classification rate (BCR):

$$BCR = \frac{1}{\#C} \sum_{i=1}^{\#C} \frac{TC_i}{C_i}$$

The legend for the formulas is as follows:

- TP is the number of true positives.
- FP is the number of false positives.
- TN is the number of true negatives.
- FN is the number of false negatives.
- $\#C$ is the number of classes.
- TC_i is the number of correct predictions for the instances of class i (testing set).
- C_i is the number of instances in class i (testing set).

C.3 Stability of feature selection

In order to evaluate the feature selection aspect of our solutions, we use a stability index in conjunction with BCR. Since the sets of features selected by the algorithm are not guaranteed to have the same size, we have chosen Lustgarten's adjusted similarity measure (ASM)[2].

The ASM is defined as the average of the pairwise similarities $S_A(s_i, s_j)$. Consequently, we can compute the pairwise similarities between all of our outer cross-validation's iterations and obtain a general stability measure by averaging them.

With n being the total number of features in the dataset, c the number of feature subsets[3] and s_i the i^{th} subset of selected features, the ASM is defined as follows:

$$S_A(s_i, s_j) = \frac{|s_i \cap s_j| - \frac{|s_i||s_j|}{n}}{\min(|s_i|, |s_j|) - \max(0, |s_i| + |s_j| - n)}$$

$$ASM = \frac{2}{c(c-1)} \sum_{i=1}^{c-1} \sum_{j=i+1}^{c} S_A(s_i, s_j)$$

[2]See main report's bibliography for reference.

[3]equal to the number of folds in the outer cross-validation

Lustgarten's *ASM* was chosen as a measure of stability over the simpler Kuncheva index for one reason: there is no guarantee that two sets of features will have the same size. Consequently, the Kuncheva index cannot be used in this case.

Undefined ASM By examining the above formula for $S_A(s_i, s_j)$, we can see that some feature subsets might result in an undefined value $\left(\frac{0}{0}\right)$. In particular, this happens when at least one of the two subsets corresponds to the full set of features. Indeed, if we assume that s_i is the full set of features, the formula becomes:

$$S_A(s_i, s_j) = \frac{|s_j| - \frac{n|s_j|}{n}}{|s_j| - (n + |s_j| - n)}$$
$$S_A(s_i, s_j) = \left(\frac{0}{0}\right)$$

When computing the ASM, we choose to omit undefined $S_A(s_i, s_j)$. It then follows that the ASM is undefined if all the $S_A(s_i, s_j)$ are undefined.

C.3.1 Computing $S_A(s_i, s_j)$ in Random Intersection Trees

Since the *Random Intersection Trees* algorithm only return interactions of features, there is no explicit feature selection occurring. This can be placed in contract to a standard filter feature selection algorithm which typically returns the set of selected features or, at least, a weight for each feature. Thankfully, it's possible to reduce a *Random Intersection Trees* algorithm to a filter feature selection.

To do so, we count the number of occurrences for each feature across all the returned interactions. The following rule can then be used:

- If a feature f occurs strictly more than C times, it is selected.
- If not, f is not selected.

For the various experiments in this thesis, $C = 0$ was used. While this may seem questionable, the fact remains that a feature only needs to occur once in the returned interactions in order to have an effect on the classification.